IN THE UNITED STATES PATENT AND TRADEMARK OFFICE

Assistant Commissioner for Patents
Washington, D.C. 20231

JC685 U.S. PTO
08/04/00

Atty. Dkt.: 550-181

Sir:

Date: August 4, 2000

JC511 U.S. PTO
09/633358
08/04/00

Attached for filing is the patent application of:
Inventor: BARTON
Entitled: **UDATING COMPUTER FILES**
and including attachments as noted below:

- ☒ Declaration, ☒ Abstract
- 20 pages of specification and claims (including 66 numbered claims), and
- 5 sheets of accompanying drawing/s.
- ☒ Record & return the attached assignment to the undersigned.
- ☐ Priority is hereby claimed under 35 USC 119 based on the following foreign applications, the entire content of which is hereby incorporated by reference in this application:

Application Number	**Country**	**Day/Month/Year Filed**

, respectively.

- ☐ Certified copy(ies) of foreign application(s) is/are attached.
- ☐ Please amend the specification by inserting before the first line --This is a of PCT application , filed , the entire content of which is hereby incorporated by reference in this application.--
- ☐ Priority is hereby claimed under 35 USC 120/365 based on the following prior PCT applications designating the U.S., the entire content of which is hereby incorporated by reference in this application:

Application Number	**Country**	**Day/Month/Year Filed**

- ☐ This application is based on the following prior provisional application(s):

Application No.	**Filing Date**

, respectively, the entire content of which is hereby incorporated by reference in this application, and priority is hereby claimed therefrom.

- ☐ Please amend the specification by inserting before the first line: -- This application claims the benefit of U.S. Provisional Application No. , filed , the entire content of which is hereby incorporated by reference in this application.
- ☐ Verified Statement attached establishing "small entity" status (Rules 9 & 27)
- ☐ The Examiner's attention is directed to the prior art cited in the parent application by applicant and/or Examiner for the reasons stated therein.
- ☐ Preliminary amendment to claims (attached hereto), to be entered before calculation of the fee below.
- ☐ Also attached:

FILING FEE IS BASED ON CLAIMS AS FILED LESS ANY HEREWITH CANCELED

Basic Filing Fee					$	690.00
Total effective claims	66	- 20 (at least 20) =	46	x $ 18.00	$	828.00
Independent claims	6	- 3 (at least 3) =	3	x $ 78.00	$	234.00
If any proper multiple dependent claims now added for first time, add $260.00 (ignore improper)					$	0.00
				SUBTOTAL	$	1752.00
If "small entity," then enter half (1/2) of subtotal and subtract					-$(	0.00)
				SECOND SUBTOTAL	$	1752.00
Assignment Recording Fee ($40.00)					$	40.00
				TOTAL FEE ENCLOSED	$	1,792.00

Any future submission requiring an extension of time is hereby stated to include a petition for such time extension.
The Commissioner is hereby authorized to charge any deficiency in the fee(s) filed, or asserted to be filed, or which should have been filed herewith (or with any paper hereafter filed in this application by this firm) to our **Account No. 14-1140.** A duplicate copy of this sheet is attached.

1100 North Glebe Road, 8th Floor
Arlington, Virginia 22201-4714
Telephone: (703) 816-4000
Facsimile: (703) 816-4100
SCS:PDC

NIXON & VANDERHYE P.C.
By Atty: Stanley C. Spooner, Reg. No. 27,393

Signature: Richard Becker 22770 (Stan)

IN THE UNITED STATES PATENT AND TRADEMARK OFFICE

APPLICATION PAPERS

OF

CHRISTOPHER ANDREW BARTON

FOR

UPDATING COMPUTER FILES

BACKGROUND OF THE INVENTION

Field of the Invention

This invention relates to the field of computers. More particularly, this invention relates to mechanisms for the updating of computer files.

Description of the Prior Art

A problem in the field of computing is the requirement for regular updating of computer files (possibly by downloading a complete new version of the file or by an incremental update in which modifying data for modifying the existing file to form the updated version is downloaded) held on many different computers. A software update may be required because the program has altered in response to the occurrence of bugs within the program or to add additional functionality to a program. Another need for frequent computer file updates is when the computer file represents rapidly evolving data needed by the computer. An example of this is the computer virus definitions data that is used by many anti-virus computer programs. This computer virus definition data is typically updated when a new virus is encountered such that the anti-virus software may provide counter-measures to the new virus. In order that the anti-virus software being used may operate in an effective manner it is important that it should use the most up to date virus definition data.

In response to this need, anti-virus software suppliers often provide download facilities from which users can download the most up to date versions of the computer virus definition data. One problem with this approach is that a user must know that an updated virus definition data file is present in order that it should be downloaded. One way to deal with this is to configure the computer program software to automatically check for new computer virus definition data at periodic intervals. If these intervals are made too short, then this presents an unnecessary burden upon the computer systems involved. Conversely, if the intervals are made too large, then a significant update required to deal with a new virus threat may not be downloaded in sufficient time to adequately protect from that virus threat.

A further problem associated with the downloading of virus definition data from the anti-virus software supplier is that peak demand for the download of the new data may cause the systems to malfunction. Computer viruses are becoming increasingly common and destructive. With this background, the release of a new computer virus attracts considerable media attention resulting in many users

simultaneously trying to download the updated data in a manner that causes this process to fail.

Various techniques for updating software are disclosed in US-A-5,940,074, US-A-4,763,271, US-A-5,919,247, US-A-5,577,244, US-A-5,809,287, US-A-5,933,647 and US-A-5,732,275. A technique for updating anti-virus DAT files via a "push" method is disclosed in US-A-6,035.423.

SUMMARY OF THE INVENTION

Viewed from one aspect the present invention provides a computer program product for updating a computer file used by a computer, said computer program product comprising:

(i) tag detecting code operable to detect within data received by said computer a tag indicative of existence of an updated version of said computer file; and

(ii) update triggering code operable upon detection of said tag to trigger downloading from a predetermined source to provide said updated version of said computer file for use by said computer.

The invention uses received data (such as e-mail messages) to provide peer-to-peer notification of the existence of an updated version of a computer file. In this way, the existence of an updated version of a computer file may be automatically disseminated in a manner that provides a degree of smoothing in the level of demand on the downloading systems for updating the computer file. Furthermore, those computers that receive more data from elsewhere are more likely to receive notification of an updated version of a computer file sooner than those computers receiving little data from elsewhere. Computers receiving much data from elsewhere are typically highly active and accordingly are the ones where early receipt of the updated computer file will be most beneficial. As an example, in the context of the distribution of computer virus definition data, computers receiving many e-mail messages are often the ones at high risk of computer virus infection and so the technique of the invention that results in these computers being likely to be among the first to be updated matches well with a priority based upon the risk from a new computer virus.

The invention recognises that with the widespread adoption of data transfer mechanisms, such as e-mail messaging, between computers, these mechanisms can be effectively used to disseminate information regarding computer file updates with very little additional overhead. In particular, in preferred embodiments the tag may be

embedded within a header portion of an e-mail message. Such header portions already include other information typically giving the transmission history of the message, which is not normally acted upon by the user and thus simplifying the embedding within this header of notification of computer file updates. This embedding may be done, for example, by any of a mail gateway, a server or workstation.

A particularly effective form of tag is one which indicates the version level of a particular computer file as this lends itself to ready comparison with other data and matches the way in which computer file suppliers already characterise their updates.

A computer receiving an e-mail message may also inserts its own tag into the message if it is itself using a computer file more up-to-date than any tag already present within the e-mail message. The computer could replace any existing tag already within the e-mail message with its indication of the more up to date version of that computer file, or alternatively it may simply add an additional tag indicating the existence of that more up to date version of the computer file. In either case, this feature significantly enhances the way in which the notification of the existence of an updated computer file will cascade through a network of many linked computers via e-mail messages, i.e. once a particular computer has itself been updated, then it may serve to add the notification of the existence of that update to any further e-mail messages that pass through it.

Whilst the present invention is suitable for the distribution of many different types of computer files, it is particularly well suited to use in anti-virus systems whereby the computer file may be a computer virus definition file, a computer virus detection engine file or an anti-virus computer program file. In each of these cases, the automatic notification of the existence of an update in a manner that can trigger the download of that update whilst avoiding excessive peaks in download demand is strongly advantageous.

The predetermined source from which a computer can download its updated version may take many forms. Within a local area network, a particular client computer may be configured to seek its updates from a specified location on a local server computer. A local server computer may be configured to seek its updates via the internet from a specified FTP source. There are numerous other possibilities.

In order to reduce the likelihood of malicious use of this updating mechanism, preferred embodiments are one in which the tag data is encrypted to make its misuse

more difficult. A further safeguard that might be used instead or as well as encryption that an update will not be triggered if the difference in the version level indicated is too large (e.g. greater than 1000 version levels) is as this may well be indicative of malicious intervention or malfunction.

Preferred embodiments also seek to smooth peak download demand by introducing an initial update delay period following detection of the tag. This update delay period can be varied such that a server version of the program will attempt to download very rapidly as the risk for a server is large whereas a workstation version may have a longer initial delay before attempting a download. The delay can be configured for each computer to match the risk associated with that computer using an out-of-date file.

Should the download fail, then the problem of an excessive build-up in requests for download may be reduced by the introduction of a failure delay period before the computer retries to download the new version of the computer file. This failure delay period may be made psuedo-random in order to try to reduce any pathological condition with competing computers simultaneously reissuing their requests for download.

Viewed from another aspect the present invention provides a computer program product comprising:

(i) tag inserting code operable to insert a tag within data indicating a version level of a computer file used by a first computer, said tag being operable to trigger an update of an older version of said computer file used by a second computer when said data is received by said second computer.

It will be appreciated that some elements within a computer network may not themselves use a particular computer file and yet may be usefully integrated within the scheme of this invention by being configured to insert suitable tags within E-mail messages passing through them.

The present invention also provides a method of updating a computer file and an apparatus for updating a computer file in accordance with the appended claims.

The above, and other objects, features and advantages of this invention will be apparent from the following detailed description of illustrative embodiments which is to be read in connection with the accompanying drawings.

BRIEF DESCRIPTION OF THE DRAWINGS

Figures 1, 2 and 3 illustrate operation of the technique of the present invention in distributing notification of the existence of an update and triggering download of an update;

Figure 4 is a flow diagram showing the processing performed by a peer computer of the type illustrated in Figures 1 to 3;

Figure 5 schematically illustrates a tag of the type that may be inserted within an e-mail message header; and

Figure 6 illustrates a computer apparatus that may be used to implement the technique of the present invention.

DESCRIPTION OF THE PREFERRED EMBODIMENTS

Figure 1 shows a plurality of computers linked via the internet 2 (or some other communication link). A software supplier provides an FTP server 4 from which updated versions of a computer file may be downloaded. Figure 1 illustrates a first local area network 6 and a second local area network 8 linked to the internet 2. The first local area network 6 includes a mail gateway computer 10, a local server 12 and two client workstations 14, 16. In a similar way the second local area network 8 includes a mail gateway 18, a local server 20 and two client workstations 22, 24.

Using known e-mail protocols and computer programs, the various workstation computers 14, 16, 22, 24 shown in Figure 1 exchange e-mail messages. In alternative embodiments the data exchange could take other forms, such as internet web pages or word processing files that contain headers or the like in which tags for triggering updates may be embedded. All of the computers illustrated in Figure 1 utilise anti-virus software that requires access to an up to date computer virus definition data file. As users of this computer virus definition data file the various mail gateway, server and workstation computers are peers. The illustrated starting situation in Figure 1 is that all of the computers are running the most up to date version (#3) of the computer file that is the version currently stored on the FTP server 4.

Figure 2 illustrates the start of the dissemination of an update. The software supplier loads an updated version of the computer virus definition data file onto the FTP server 4. A manual or an automatic timed update of the mail gateway 10 then takes place that updates the version of the computer file on the mail gateway to #4. Subsequently, an e-mail message is issued by the workstation computer 16 destined for the workstation computer 24. The anti-virus software within the workstation

computer 16 tags the e-mail header of the e-mail message with the version #3 of the computer virus definition data file that the workstation computer 16 is currently using. This e-mail message first passes through the local server 12 which is also using this same version #3 and so leaves the tag unaltered. When the e-mail message reaches the mail gateway 10, the mail gateway checks the tag within the e-mail message header and determines that it is itself using a more up-to-date version of the computer file in question and so replaces the tag with one that indicates that version #4 has been used and is available. The tag may also indicate whether or not that computer may itself serve as the source of the update data for other computers. This e-mail message then propagates via the internet 2 and through the mail gateway 18 and the local server 20 until it reaches the workstation computer 24. Each of the mail gateway 18, the local server 20 and the workstation computer 24 examines the tag within the e-mail message header and determines that it indicates the existence of a more up to date version of the computer virus definition data file than that which they are currently using. Accordingly, each of the mail gateway 18, the local server 20 and the workstation computer 24 is triggered (after an initial delay period) to download the updated version of the computer file from the FTP server 4 (a predetermined source of updates). The "*" indicates that a particular peer computer has detected that it should download an updated version of the computer file. The mail gateway 18, the local server 20 and the workstation computer 24 will continue to use version #3 until they have the updated version #4. The update mechanism used is preferably the pre-existing standard "pull" mechanism, but it is envisaged that alternative emergency or special purpose update mechanisms for use when triggered by a received tag could be provided.

Figure 3 shows the situation when the updates to the mail gateway 18, the local server 20 and the workstation computer 24 have all taken place. In the example shown, the user of the workstation computer 24 sends an e-mail message to the user of the workstation computer 16 that is also copied to the user of the workstation computer 22. As the workstation computer 24 that is the source of this e-mail message has now been updated to version #4, it includes within the header of the e-mail message a tag indicating that version #4 exists. When the workstation computer 22 receives this message, this is detected as indicating that the workstation computer 22 should download the updated version #4 from the FTP server 4. The e-mail message also propagates via the local server 20, the mail gateway 18 and the mail

gateway 10 towards the other target recipient that is the workstation computer 16. All of these peer computers have already been updated, thus they do not action version #4 tags and below. The first computer reached in this transmission path that has not yet been updated is the local server 12 and the second is the workstation computer 16. Both of these computers also detect that the tag shows a version level #4 higher than that which they are currently using #3 and accordingly trigger the download of an update from the FTP server 4.

It will be appreciated that the mechanisms shown in Figures 1, 2 and 3 allow for the automatic and rapid dissemination of the information that an updated file exists. Furthermore, those computers that receive more e-mail messages are more likely to receive this notification sooner. These are the very computers that are generally at a high risk from computer viruses and accordingly it is appropriate that they should be the first that download the updated version of the computer virus definition data file. A little used computer will only download its update at a later time, and yet this will pose potentially lower level of risk since the little used computer is less likely to receive an infectious element.

Figure 4 is a flow diagram schematically illustrating the processing performed by a particular peer computer.

At step 26, the computer receives an e-mail message. At step 28 the computer searches the message header of the e-mail for any header tag present and decrypts this if it is found. In some embodiments the header tag may not be encrypted.

At step 30, a test is made as to whether any header tag has been found. If a header tag has not been found, then the e-mail message is scanned for computer viruses using the existing anti-virus software at its current level of update and a header tag added to the e-mail message indicative of that current level of update at step 32.

If a header tag is found at step 30, then step 34 tests whether or not that header tag indicates a version of the software that is older than that held by the computer performing the process illustrated in Figure 4. If the received header tag is indicative of an older version, the processing proceeds to step 32 at which the e-mail message is scanned using what is known to be more up to date data than has previously been applied to that message and the header tag indicative of that more up to date data is added to the e-mail header. The existing tag indicative of the older version of the computer file may or may not be removed. The tag may also include parameters

indicative of previous processing applied to the message, such as the program options set (e.g. all files, macro heuristics, all heuristics) on previous anti-virus scans applied to the message. These parameters can be used to determine whether or not further scanning is to be applied by the computer currently processing the message.

If the test at step 34 indicates that the received e-mail message included a header tag that was not older than the local version, then processing proceeds to step 36. Step 36 tests whether or not the tag of the received e-mail message indicates a newer version of the computer file is available. If a newer version is not available, then processing proceeds to step 37. Step 37 decided whether or not the message should be scanned at step 32 in dependence upon parameters set on the processing computer and parameters within the tag as mentioned above that indicate in more detail what previous scanning has been applied to the message. If the test at step 36 indicates that a newer version of the computer file than that stored by the local computer is available, then processing proceeds to step 40.

Step 40 tests how may versions ahead of the current version the tag within the received e-mail message indicates is available. If this number exceeds a predetermined threshold N, then this is indicative of some malfunction or malicious interference with the message tags and accordingly processing proceeds to step 32.

If the test at step 40 indicates that the updated version is less than the threshold number N ahead of the currently used version, then processing proceeds to step 42 which scans using the currently held version and then imposes an initial delay before processing proceeds to step 44 where an update attempt is triggered to download the updated version of the computer file from a remote source. The remote source may be an FTP server 4 linked via the internet 2 (or any other link) to the computer in question, or could be a server file location within a local area network 6, 8 or some other source.

At step 46, a test is made as to whether or not the update attempt has failed. If the update attempt has not failed, then processing continues with other normal e-mail processing operations at step 38.

If the update attempt has failed, then processing proceeds to step 48 which imposes a psuedo-random failure delay prior to returning processing to step 44 to attempt another update. A predetermined number of update attempt failures may trigger a user warning message.

Figure 5 illustrates a message tag of the type that may be inserted within an E-mail message header. The "X-" prefix in the tag is one defined some in standard e-mail protocols as indicating that the information that follows on the line is for information purposes only and is not actively processed in normal systems. In the illustrated example, the tag starts with a coding for "McAfee-Sig:". This is a code sequence that is searched for by peer computers within e-mail message headers to detect the presence of a tag indicating what version levels of an anti-virus system have already been applied to that e-mail message. This version information follows in the form of "<S#-xxxE#-yyyD#-zzz>", portions of which respectively indicate the anti-virus software program version number, the computer virus detection engine version number and the computer virus definition data version number. In practice, this version information may be encrypted to make it more difficult to tamper with the information in an attempt to misdirect or interfere with the update mechanisms. Various known encryption techniques may be used. The tag could also include parameters indicating previously applied scan options or other data and could extend over more than one line.

Figure 6 schematically illustrates a computer 50 of the type that may be used to implement the techniques described above (more simple appliance type devices could also be used). The computer 50 includes a central processing unit 52, a read only memory 54, a random access memory 56, a network link 58, a hard disk drive 60, a display driver 62, a display 64, a user input/output driver 64, a keyboard 66 and a mouse 68.

In operation, the central processing unit 52 executes computer programs stored upon the hard disk drive 60 or within the read only memory 54 using the random access memory as working memory. User inputs for controlling the computer 50 are received from the keyboard 66 and the mouse 68 via the user input/output unit 64. Processing results may be displayed to the user using the display 64 via the display driver unit 62.

In operation, an e-mail message may be received from the internet 2 via the network link 58 into an e-mail program being executed by the central processing unit 52. Part of this e-mail program may trigger an anti-virus scan of the received e-mail. This anti-virus scan includes the processing illustrated in Figure 4 and accordingly detects if the received e-mail message includes the information that an updated version of any of the components of the anti-virus system exists for download.

Should such updated versions exist, then they may be downloaded by the computer 50 under program control from a remote source, such as an FTP server 4, via the internet 2 and the network link 58.

The updated computer files, such as the anti-virus software program, the search engine program or the virus definitions, will typically then be stored on the hard disk drive 60 of the computer 50. The program that causes the processing of Figure 4 to take place will also typically be stored upon the hard disk drive 60. The computer program may be distributed via a recordable medium, such as a floppy disk or a CD, or may itself be downloaded via the network link 58.

The computer 50 may pass the e-mail message onto another computer or may itself originate a new e-mail message. In either case, any outbound e-mail message is marked with a tag indicating the version levels of the anti-virus software components used by the computer 50 if these are more up to date than any indications already within the e-mail message.

Although illustrative embodiments of the invention have been described in detail herein with reference to the accompanying drawings, it is to be understood that the invention is not limited to those precise embodiments, and that various changes and modifications can be effected therein by one skilled in the art without departing from the scope and spirit of the invention as defined by the appended claims.

I CLAIM:

1. A computer program product for updating a computer file used by a computer, said computer program product comprising:

(i) tag detecting code operable to detect within data received by said computer a tag indicative of existence of an updated version of said computer file; and

(ii) update triggering code operable upon detection of said tag to trigger downloading from a predetermined source to provide said updated version of said computer file for use by said computer.

2. A computer program product as claimed in claim 1, wherein said tag is embedded within a header portion of said data.

3. A computer program product as claimed in claim 1, wherein said data is an e-mail message.

4. A computer program product as claimed in claim 1, wherein said predetermined source provides one of said updated version of said computer file or modifying data for modifying said computer file currently used by said computer to form said updated version of said computer file.

5. A computer program product as claimed in claim 1, wherein said tag includes a version identifier and further comprising version comparing code operable to compare said version identifier with data indicative of a version level of said computer file to determine if an updated version of said computer file exists.

6. A computer program product as claimed in claim 1, further comprising tag inserting code operable to insert into data being processed by said computer a tag indicative of a version level of said computer file currently used by said computer.

7. A computer program product as claimed in claim 6, wherein said tag inserting code does not insert a tag if said data already includes a tag indicative of the existence of said version level of said computer file currently used by said computer or of the existence of a more recent version level of said computer file used by said computer.

8. A computer program product as claimed in claim 1, wherein said computer file is a computer virus definition data file.

9. A computer program product as claimed in claim 1, wherein said computer file is a computer virus detection engine program file.

10. A computer program product as claimed in claim 1, wherein said computer file is an anti-virus computer program file.

11. A computer program product as claimed in claim 1, wherein said tag includes data indicative of a version level of one or more of:

(i) a computer virus definition data file;

(ii) a computer virus detection engine program file; and

(iii) an anti-virus computer program file.

12. A computer program product as claimed in claim 1, wherein said tag includes parameters indicative of one or more processing operations previously performed upon said data.

13. A computer program product as claimed in claim 12, wherein said one or more processing operations are anti-virus scanning operations.

14. A computer program product as claimed in claim 1, wherein said predetermined source is contains an updated version of said computer file provided by a supplier of said computer file.

15. A computer program product as claimed in claim 1, wherein said predetermined source is remote from said computer.

16. A computer program product as claimed in claim 11, wherein said predetermined source is accessed via an internet link.

17. A computer program product as claimed in claim 1, wherein said tag data is encrypted tag data and said tag detecting code includes decryption code for decrypting said encrypted tag data.

18. A computer program product as claimed in claim 1, wherein said update triggering code waits for an initial update delay period following detection of said tag before starting download of said updated version of said computer file.

19. A computer program product as claimed in claim 1, wherein if download of said updated version of said computer file fails, then said update triggering code waits for a failure delay period before retriggering downloading of said updated version of said computer file.

20. A computer program product as claimed in claim 15, wherein said failure delay period is a psuedo-random value determined by said update triggering code.

21. A computer program product as claimed in claim 1, wherein said update triggering code does not trigger an update if said tag indicates that said updated version of said computer file is more than a predetermined number of versions of ahead of said computer file currently used by said computer.

22. A computer program product comprising:

(i) tag inserting code operable to insert a tag within data indicating a version level of a computer file used by a first computer, said tag being operable to trigger an update of an older version of said computer file used by a second computer when said data is received by said second computer.

23. A method of updating a computer file used by a computer, said method comprising the steps of:

(i) detecting within data received by said computer a tag indicative of existence of an updated version of said computer file; and

(ii) upon detection of said tag, triggering downloading from a predetermined source to provide said updated version of said computer file for use by said computer.

24. A method as claimed in claim 23, wherein said tag is embedded within a header portion of said data.

25. A method as claimed in claim 23, wherein said data is an e-mail message.

26. A method as claimed in claim 23, wherein said predetermined source provides one of said updated version of said computer file or modifying data for modifying said computer file currently used by said computer to form said updated version of said computer file.

27. A method as claimed in claim 23, wherein said tag includes a version identifier and further comprising the steps of comparing said version identifier with data indicative of a version level of said computer file to determine if an updated version of said computer file exists.

28. A method as claimed in claim 23, further comprising the step of inserting into data being processed by said computer a tag indicative of a version level of said computer file currently used by said computer.

29. A method as claimed in claim 28, wherein a tag is not inserted if said data already includes a tag indicative of the existence of said version level of said computer file currently used by said computer or of the existence of a more recent version level of said computer file used by said computer.

30. A method as claimed in claim 23, wherein said computer file is a computer virus definition data file.

31. A method as claimed in claim 23, wherein said computer file is a computer virus search engine program file.

32. A method as claimed in claim 23, wherein said computer file is an anti-virus computer program file.

33. A method as claimed in claim 23, wherein said tag includes data indicative of a version level of one or more of:

(i) a computer virus definition data file;

(ii) a computer virus detection engine program file; and

(iii) an anti-virus computer program file.

34. A method as claimed in claim 23, wherein said tag includes parameters indicative of one or more processing operations previously performed upon said data.

35. A method as claimed in claim 34, wherein said one or more processing operations are anti-virus scanning operations.

36. A method as claimed in claim 23, wherein said predetermined source is contains an updated version of said computer file provided by a supplier of said computer file.

37. A method as claimed in claim 23, wherein said predetermined source is remote from said computer.

38. A method as claimed in claim 37, wherein said predetermined source is accessed via an internet link.

39. A method as claimed in claim 23, wherein said tag data is encrypted tag data and said step of detecting includes decrypting said encrypted tag data.

40. A method as claimed in claim 23, further comprising the step of waiting for an initial update delay period following detection of said tag before starting download of said updated version of said computer file.

41. A method as claimed in claim 23, wherein if download of said updated version of said computer file fails, then further comprising waiting for a failure delay period before retriggering downloading of said updated version of said computer file.

42. A method as claimed in claim 41, wherein said failure delay period is a psuedo-random value determined by said update triggering code.

43. A method as claimed in claim 23, wherein an update is not triggered if said tag indicates that said updated version of said computer file is more than a predetermined number of versions of ahead of said computer file currently used by said computer.

44. A method of operating a computer comprising the step of:

(i) inserting a tag within data indicating a version level of a computer file used by a first computer, said tag being operable to trigger an update of an older version of said computer file used by a second computer when said data is received by said second computer.

45. Apparatus for updating a computer file used by a computer, said apparatus comprising:

(i) tag detecting logic operable to detect within data received by said computer a tag indicative of existence of an updated version of said computer file; and

(ii) update triggering logic operable upon detection of said tag to trigger downloading from a predetermined source to provide said updated version of said computer file for use by said computer.

46. Apparatus as claimed in claim 44, wherein said tag is embedded within a header portion of said data.

47. Apparatus as claimed in claim 44, wherein said data is an e-mail message.

48. Apparatus as claimed in claim 44, wherein said predetermined source provides one of said updated version of said computer filer or modifying data for modifying said computer file currently used by said computer to form said updated version of said computer file.

49. Apparatus as claimed in claim 45, wherein said tag includes a version identifier and further comprising version comparing logic operable to compare said

version identifier with data indicative of a version level of said computer file to determine if an updated version of said computer file exists.

50. Apparatus as claimed in claim 45, further comprising tag inserting logic operable to insert into data being processed by said computer a tag indicative of a version level of said computer file currently used by said computer.

51. Apparatus as claimed in claim 50, wherein said tag inserting logic does not insert a tag if said data already includes a tag indicative of the existence of said version level of said computer file currently used by said computer or of the existence of a more recent version level of said computer file used by said computer.

52. Apparatus as claimed in claim 45, wherein said computer file is a computer virus definition data file.

53. Apparatus as claimed in claim 45, wherein said computer file is a computer virus detection engine program file.

54. Apparatus as claimed in claim 45, wherein said computer file is an anti-virus computer program file.

55. Apparatus as claimed in claim 45, wherein said tag includes data indicative of a version level of one or more of:

(i) a computer virus definition data file;

(ii) a computer virus detection engine program file; and

(iii) an anti-virus computer program file.

56. Apparatus as claimed in claim 45, wherein said tag includes parameters indicative of one or more processing operations previously performed upon said data.

57. Apparatus as claimed in claim 56, wherein said one or more processing operations are anti-virus scanning operations.

58. Apparatus as claimed in claim 45, wherein said predetermined source is contains an updated version of said computer file provided by a supplier of said computer file.

59. Apparatus as claimed in claim 45, wherein said predetermined source is remote from said computer.

60. Apparatus as claimed in claim 45, wherein said predetermined source is accessed via an internet link.

61. Apparatus as claimed in claim 45, wherein said tag data is encrypted tag data and said tag detecting logic includes decryption logic for decrypting said encrypted tag data.

62. Apparatus as claimed in claim 45, wherein said update triggering logic waits for an initial update delay period following detection of said tag before starting download of said updated version of said computer file.

63. Apparatus as claimed in claim 45, wherein if download of said updated version of said computer file fails, then said update triggering logic waits for a failure delay period before retriggering downloading of said updated version of said computer file.

64. Apparatus as claimed in claim 63, wherein said failure delay period is a psuedo-random value determined by said update triggering logic.

65. Apparatus as claimed in claim 45, wherein said update triggering logic does not trigger an update if said tag indicates that said updated version of said computer file is more than a predetermined number of versions of ahead of said computer file currently used by said computer.

66. Apparatus for inserting a tag within data, said apparatus comprising:

(i) tag inserting logic operable to insert a tag within data indicating a version level of a computer file used by a first computer, said tag being operable to

trigger an update of an older version of said computer file used by a second computer when said e-mail is received by said second computer.

ABSTRACT OF THE DISCLOSURE
UPDATING COMPUTER FILES

A computer file update distribution technique uses tags embedded within data (such as e-mail message headers) to distribute notification of the existence of more up to date versions of a computer file to peer receivers (such as mail gateways, servers and workstations) of that data that are then triggered to download those more up to date versions. More than one peer along the transmission path of the data will be triggered to perform the update. The computer files may be components of an anti-virus system. The tags may be encrypted to increase security. The tags could include parameters indicative of processing operations (e.g. anti-virus scan options) already applied to the data that in turn influence the processing performed by the receiver. The tag can also indicate whether or not the peer that sent the message would be a source for the update.

[Figure 4]

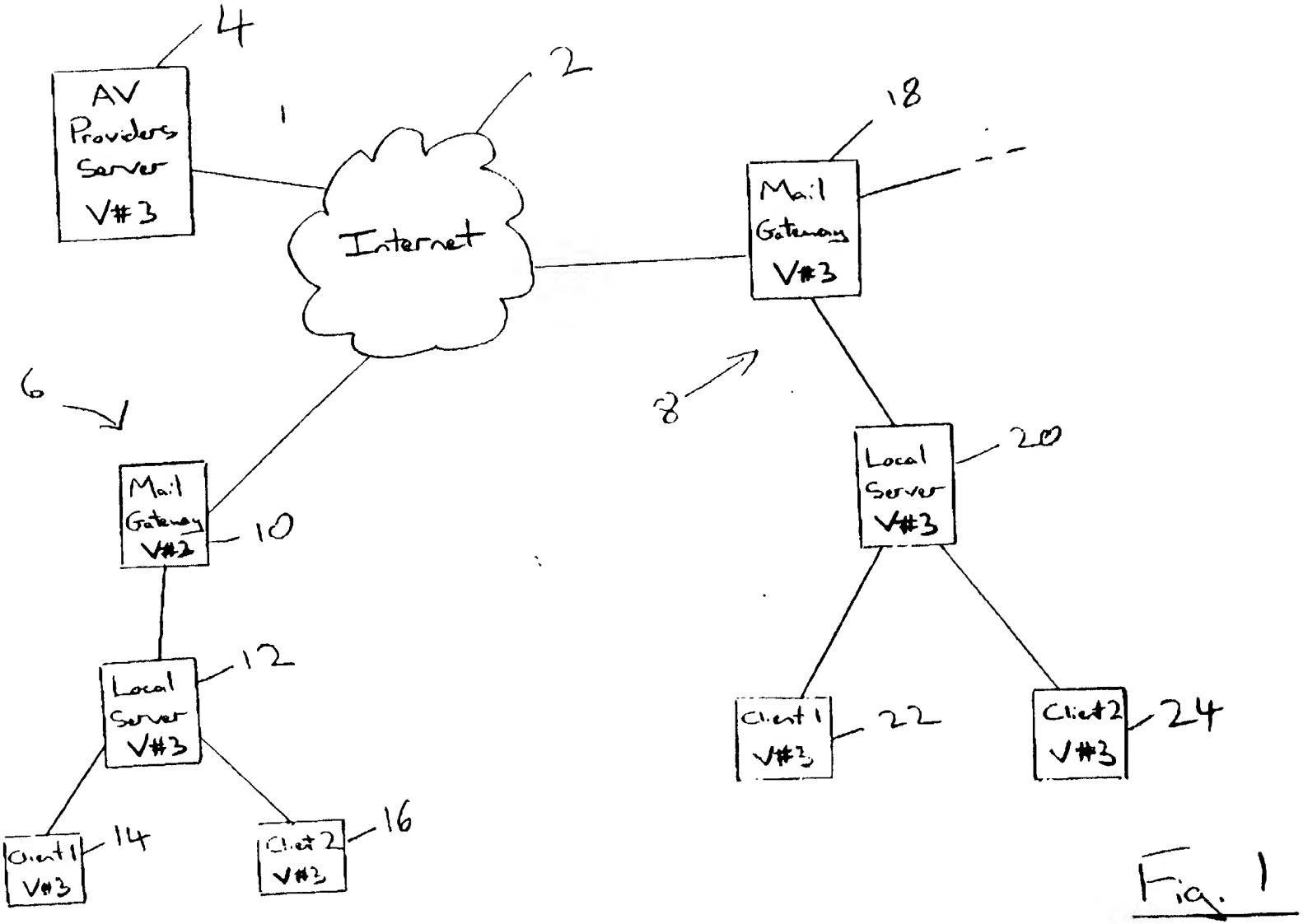

Fig. 1

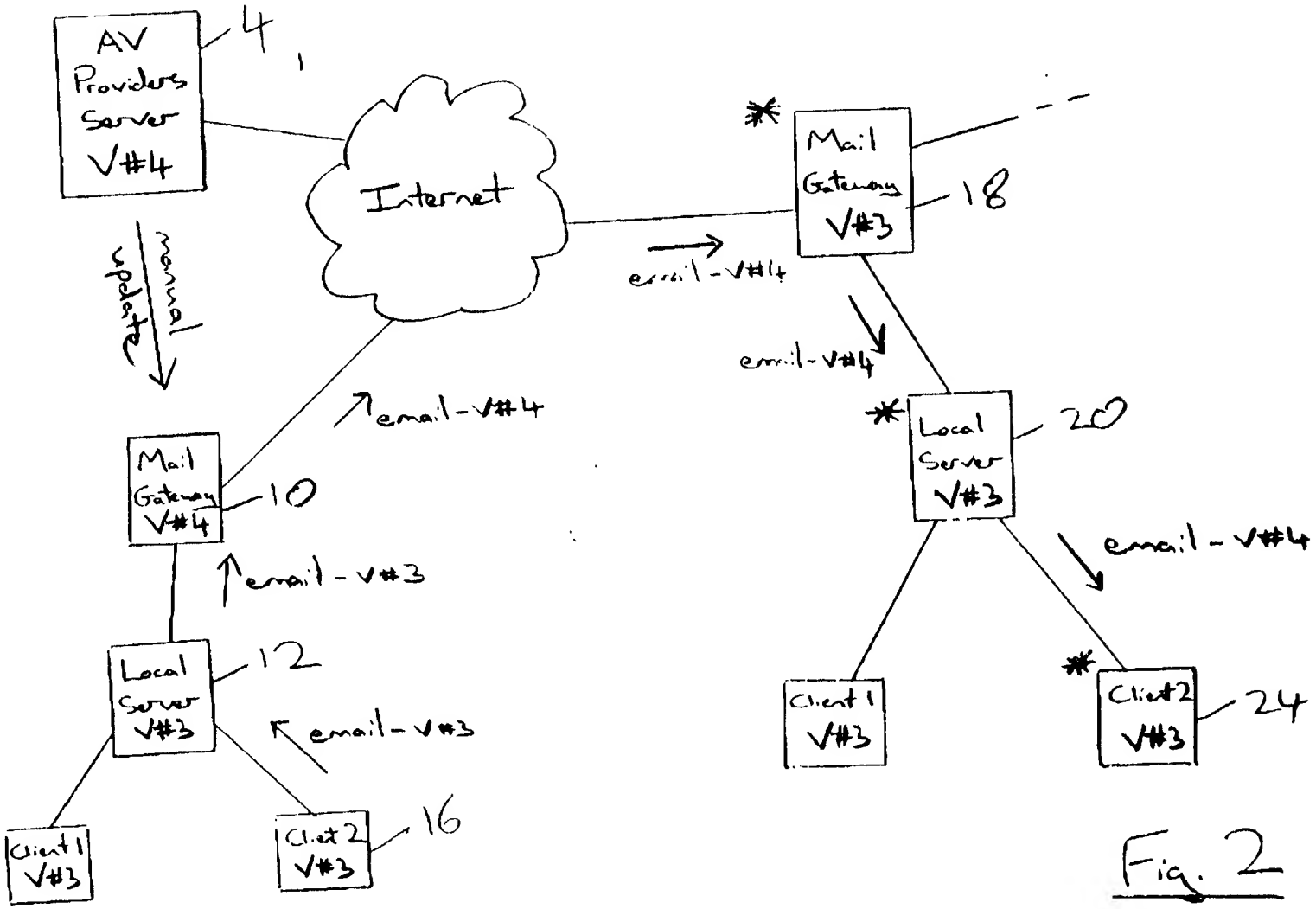
AV Providers Server V#4
4
Internet
update
manual
Mail Gateway V#4
10
email - V#4
email - V#3
Local Server V#3
12
email - V#3
Client 1 V#3
Client 2 V#3
16
Mail Gateway V#3
18
email - V#4
email - V#4
Local Server V#3
20
email - V#4
Client 1 V#3
Client 2 V#3
24

Fig. 2

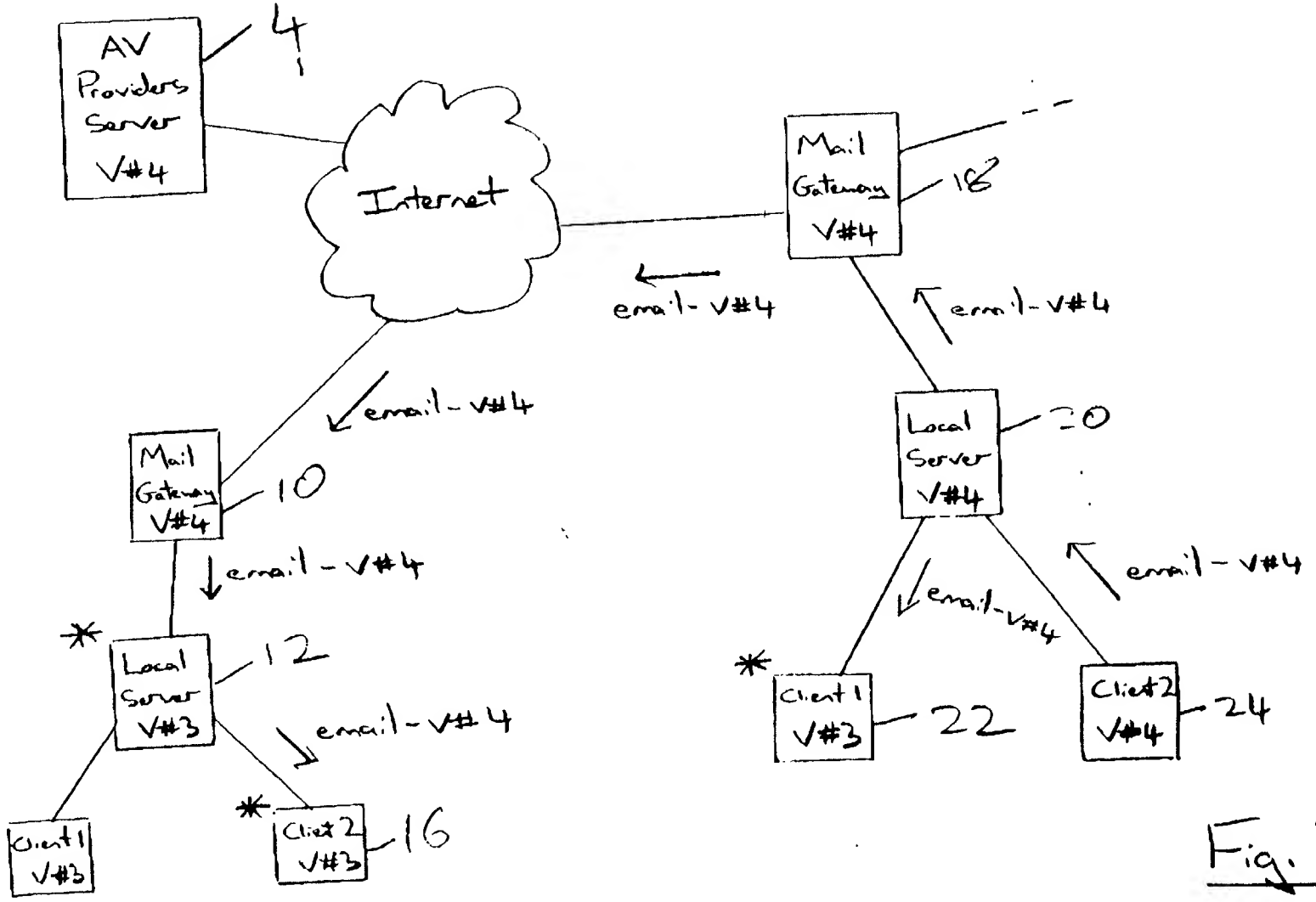

Fig. 3

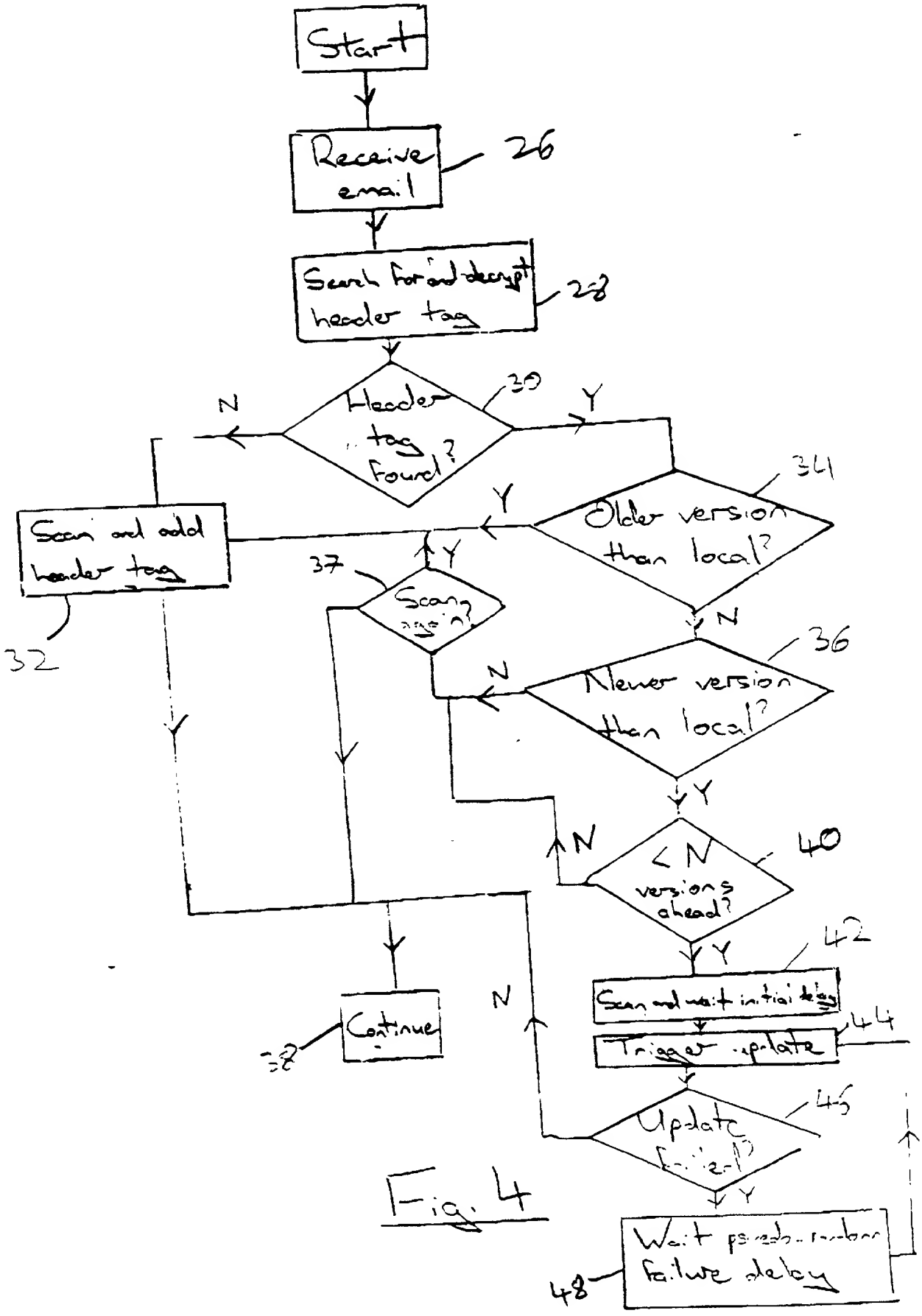
Start
Receive email
26
Search for and decrypt header tag
28
Header tag found?
30
N
Y
Scan and add header tag
32
Y
Older version than local?
34
Y
Scan again?
37
N
Newer version than local?
36
N
Y
< N versions ahead?
40
N
Y
Continue
38
Scan and wait initial delay
42
Trigger update
44
Update failed?
46
N
Y
Wait pseudo-random failure delay
48

Fig. 4

X- McAfee-sig: <S#=xxxE#=yyyO#-zzz>

Fig. 5

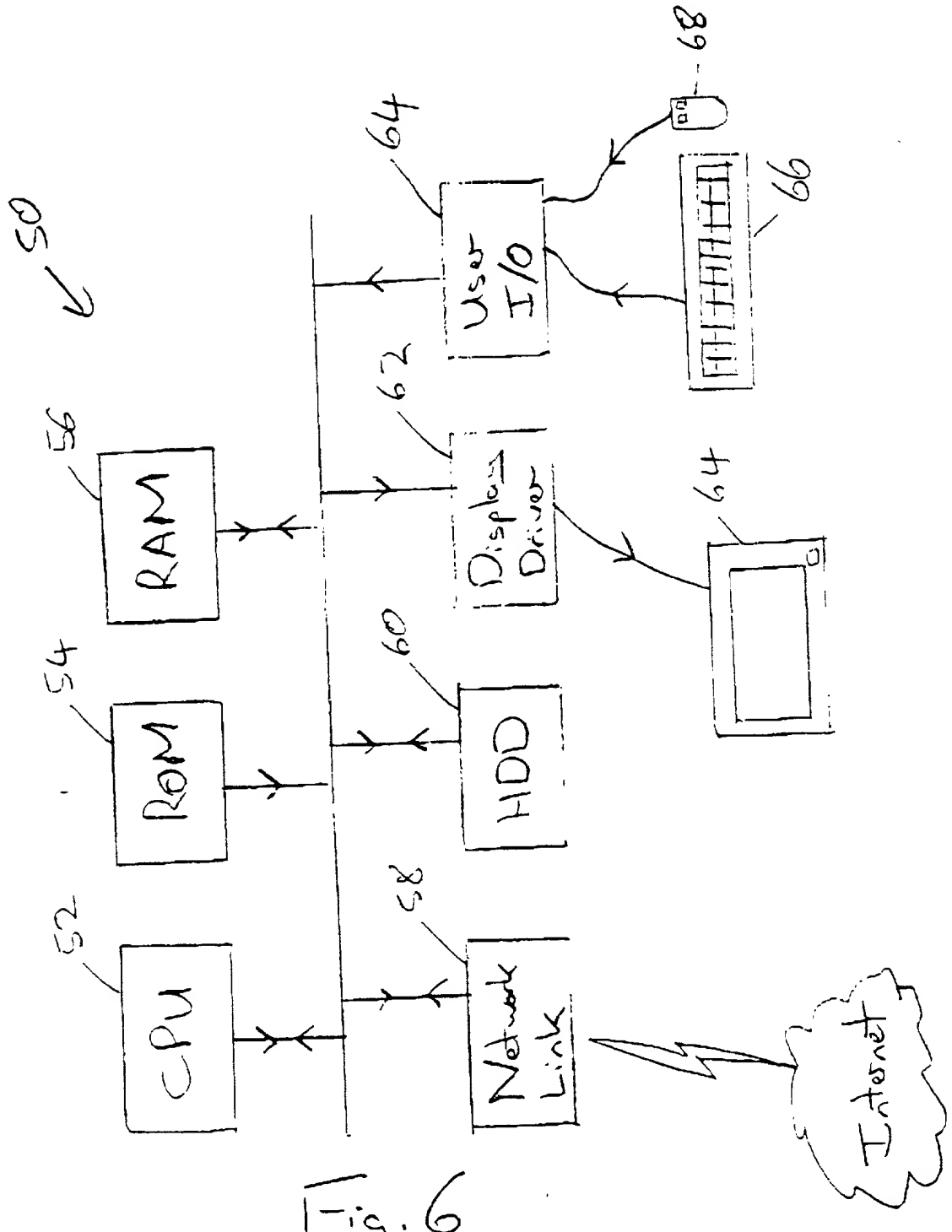

Fig. 6

P009334US

RULE 63 (37 C.F.R. 1.63)
DECLARATION AND POWER OF ATTORNEY
FOR PATENT APPLICATION
IN THE UNITED STATES PATENT AND TRADEMARK OFFICE

As a below named inventor, I hereby declare that my residence, post office address and citizenship are as stated below next to my name, and I believe I am the original, first and sole inventor (if only one name is listed below) or an original, first and joint inventor (if plural names are listed below) of the subject matter which is claimed and for which a patent is sought on the invention entitled:

UPDATING COMPUTER FILES

the specification of which (check applicable box(s)):

[X] is attached hereto
[ ] was filed on _____
Application Serial No. _____
and was amended on _____ (if applicable)

I hereby state that I have reviewed and understand the contents of the above-identified specification, including the claims, as amended by any amendment referred to above. I acknowledge the duty to disclose information which is material to the patentability of this application in accordance with 37 C.F.R. 1.56. I hereby claim foreign priority benefits under 35 U.S.C. 119/365 of any foreign application(s) for patent or inventor's certificate listed below and have also identified below any foreign application for patent or inventor's certificate having a filing date before that of the application on which priority is claimed or, if no priority is claimed, before the filing date of this application:

Prior Foreign Application(s)

			Priority Claimed	
(Number)	(Country)	(Day/Month/Year Filed)	[ ] Yes	[X] No

I hereby claim the benefit under 35 U.S.C §119(e) of any United States provisional application(s) listed below.

(Application No.) (Filing Date)

I hereby claim the benefit under 35.U.S.C. 120/365 of all prior United States and PCT international applications listed above or below and, insofar as the subject matter of each of the claims of this application is not disclosed in such prior applications in the manner provided by the first paragraph of 35 U.S.C 112, I acknowledge the duty to disclose material information as defined in 37 C.F.R. 1.56 which occurred between the filing date of the prior applications and the national or PCT international filing date of this application:

(Prior U.S./PCT Application(s)) (Application Serial No.)	(Filing Date)	(Status)(patented, pending, abandoned)

I hereby declare that all statements made herein of my own knowledge are true and that all statements made on information and belief are believed to be true; and further that these statements were made with the knowledge that wilful false statements and the like so made are punishable by fine or imprisonment, or both, under Section 1001 of Title 18 of the United States Code and that such wilful false statements may jeopardize the validity of the application or any patent issued thereon. And I hereby appoint **NIXON & VANDERHYE P.C., 1100 North Glebe Road., 8th Floor, Arlington, VA 222014714, telephone number (703) 816-4000 (to whom all communications are to be directed)**, and the following attorneys thereof (of the same address) individually and collectively my attorneys to prosecute this application and to transact all business in the Patent and Trademark office connected therewith and with the resulting patent: Arthur R Crawford, 25327; Larry S Nixon, 25640; Robert A Vanderhye, 27076; James T Hosmer, 30184; Robert W Faris, 31352, Richard G Besha, 22770; Mark E Nusbaum, 32348; Michael J Keenan, 32106; Bryan H Davidson, 30251; Stanley C Spooner, 27393. Leonard C Mitchard, 29009; Duane M Byers, 33363; Jeffry H Nelson, 30481; John R Lastova, 33149; H Warren Burnam, Jr., 29366; Thomas E Byrne, 32205, Mary J Wilson, 32955; J Scott Davidson, 33489; Alan M Kagen, 36178, William J Griffin, 31260; Robert A Molan, 29834; B J Sadoff, 36663; James D Berquist, 34776; Updeep S Gill, 37334

Inventor's signature _____ Dated 20th June 2000

Full name of sole or first inventor: Christopher Andrew Barton

Residence: United Kingdom Citizenship: United Kingdom

Post Office Address: 8 Sycamore Leys, Steeple Claydon, Buckingham, Buckinghamshire. MK18 2RH